An Experimental Comparative Study for Interactive Evolutionary Computation Problems

Yago Sáez[1], Pedro Isasi[1], Javier Segovia[2], and Asunción Mochón[3]

[1] Universidad CARLOS III de Madrid, Leganés 28911, Spain
yago.saez@uc3m.es
http://et.evannai.inf.uc3m.es/personal/ysaez/
[2] Universidad Politécnica de Madrid, Facultad de Informática, Campus Montegancedo, Boadilla del Monte 28040, Spain
[3] Universidad Nacional de Educación a Distancia, Departamento de Economía, Aplicada, Madrid 28040, Spain

Abstract. This paper presents an objective experimental comparative study between four algorithms: the Genetic Algorithm, the Fitness Prediction Genetic Algorithm, the Population Based Incremental Learning algorithm and the purposed method based on the Chromosome Appearance Probability Matrix. The comparative is done with a non subjective evaluation function. The main objective is to validate the efficiency of several methods in Interactive Evolutionary Computation environments. The most important constraint of working within those environments is the user interaction, which affects the results adding time restrictions for the experimentation stage and subjectivity to the validation. The experiments done in this paper replace user interaction with several approaches avoiding user limitations. So far, the results show the efficiency of the purposed algorithm in terms of quality of solutions and convergence speed, two known keys to decrease the user fatigue.

1 Introduction

Evolutionary Computation (EC) encompasses computational models which follow a biological evolution metaphor. The success of these techniques is based on the maintenance of genetic diversity, for which it is necessary to work with large populations. The population size that guarantees an optimal solution in a short time has been a topic of intense research [2], [3]. Large populations generally converge to better solutions, but they require more computational cost and memory requirements. Goldberg et al. [4] developed the first population-sizing equation based on the variance of fitness. They further enhanced the equation which allows accurate statistical decision making among competing building blocks (BBs) [2]. Extending the decision model presented in [2], Harik et al. [3] tried to determine an adequate population size which guarantees a solution with the desired quality. To show the real importance of the population size in Evolutionary Algorithms (EAs) He and Yao [5] showed that the introduction of a non random population decreases convergence time. However, it is not always possible to deal with such large populations, for example, when the adequacy values must be

estimated by a human being (Interactive Evolutionary Computation, IEC). Those environments require methods capable to perform well with a short number of individuals (micropopulations).

IEC techniques can be applied to different fields, such as digital treatment of images [6], composition of musical works [7], automotive design [8], automatic design of figures [9], general artistic design [10], [11], for example, the design of sculptures [12], or generation of gestures in 3D figures, aimed at virtual worlds, pictures or games, [13]. A more detailed sort of examples can be found in a complete survey about IEC [14].

The main problem in most of the above-mentioned references is that the selection criteria is based on a merely artistic and personal point of view. In these kind of applications the problems become more complex since the criteria used is subjective, (it is based on human perception and changes according to the user's opinions or personal preferences). To avoid this problem a typical IEC design problem with a non-subjective evaluation function is presented. This environment allows to run a large number of experiments and to analyze the behaviour of the four selected algorithms under the same conditions.

The remainder of the article is organized in the following way: a brief description of the algorithms used to compare the performance of our proposal is presented in section 2. Section 3 reports the description of the problem environment and the results of the comparative tests. Finally, the conclusions are drawn in section 4.

2 Description of Algorithms

IEC algorithms are difficult to test because they require the evaluation of the user in their experiments. In addition, if the designer wants to compare the new algorithms with existing ones, even more experimentation with humans is required. The proposed algorithm (Chromosome Appearance Probability Matrix, CAPM) is compared in every domain with the classical Genetic Algorithm (Simple Genetic Algorithm, SGA) with the Fitness Predictive Genetic Algorithm (FPGA) and a probabilistic algorithm called Population Based Incremental Learning (PBIL). The SGA is always a good point of reference for the comparison and the FPGA is one of the latest proposals for IEC problems [15]. The probabilistic approach gives another interesting point of view to compare with, because it is the starting point for the Estimation of Distribution Algorithms (EDA), [16] and [17]. Besides, the proposed method was partially inspired by the EDAs.

The representation chosen for all the chromosomes is the same in all the contrasted algorithms. Each individual is made up of various chromosomes which are in turn, made up of a vector of integers. In the following section the selected algorithms, except the SGA, will be briefly explained.

2.1 Fitness Predictive Genetic Algorithm

Since the user is present during the evaluation and the selection process in IEC techniques it is necessary to work with micropopulations which are easily eval-

uated. Nevertheless A small number or a reduced population affects negatively the productivity of the GAs and, as a result, improvements over the conventional GA are proposed. These improvements depend on the size of the population and the fitness prediction of all the individuals which are not evaluated by the user, [15], [27]. Thus it deals with an algorithm of M individuals and only a subset of N are shown to the user. Then the user makes its personal evaluation. Finally, the fitness of the rest (M-N) is obtained according to the differences encountered with the selection made by the user in the previous iteration. There are several types of approaches in order to estimate the fitness of the rest, but only the most simple has been developed for this paper: the Euclidean distance approach. The FPGA follows the next steps:

Step 1 - Initiate the population with N number of possible random solutions: during this process N number of individuals are randomly initiated from the population. The value of N depends on the quantity of individuals which want to be shown to the user per iteration and it is also linked to the type of problem which wants to be solved. The experiments done have been based on N=10.

Step 2 - Evaluate N candidates: this process is responsible for the evaluation of N visible candidates just as the user would do. As a result, and if decoding was necessary, each of the genotypes are converted to corresponding phenotypes. With the fitness value or adaptation of each individual of the whole N assigned, the determined evaluation function for each problem is applied. At this point, all other individuals of the population (M-N) remain to be evaluated. In the experimental domain populations of 100 individuals have been used, out of which the user evaluates the best 10, M=100 and N=10.

Step 3 - Forecast fitness values of candidates M-N: this process is used if the condition which guarantees the number of iterations is greater than 2. Alternatively, it is used when an initial reference evaluation has been made which allows the forecasting of the fitness values of the rest of the population (M-N). The necessary modifications to adapt the algorithm to the selection method have been done carefully and it corresponds to the proposals regarding to the algorithm put forward by Hsu, [15]. Predictive fitness is obtained by the calculation of Euclidean distance between the referenced chromosomes (those selected by the user and those of the individuals not evaluated by the user). FPGAs are effective in typical IEC problems in which the parameters are coded in such a way that the Euclidean distance gives an idea of how close or far an individual is from the one selected by the user. As the FPGA is not designed to be applied to numerical optimization problems and is not foreseen as being used with the proposed coding, it is possible that due to their genotype differences, two very close variables (as far as the phenotype is concerned) could produce great distances.

Step 4 - Select two parents: as with the SGA, the selection operator implemented for the experimentation is based on the selection of the two best individuals according to their adaptation or fitness value. However, the selection can only be done with the best 'N' individuals of the population. This previously mentioned limitation is typical of FPGAs and IEC techniques in which the user evaluates. Therefore, the selection forces the evaluation to a subset of the population 'N'. During each iteration, the selections made by the user are stored in the ps_1 and ps_2 variables with the intention of doing other calculations which will obtain the predictive fitness of the following iterations.

Step 5 - Cross pairs with parents: this process is exactly the same as the SGA. Once the best two individuals of fitness value are selected the cross operator is applied in order to obtain the new generation of individuals. The propagation strategy is elitist and, as a result, the selected parents go directly to the following generation. The remainder are generated from the equally probable crossing of the parents.

Step 6 - Mutate the obtained descendants: as with the SGA, this process is responsible for mutating, with a certain probability, ($P_{mutation}$) several genes of the generated individuals. The aim is to guarantee the appearance of new characteristics in the following populations and to maintain enough genetic diversity.

Step 7 - Repeat until the final condition: like the SGA, the stop condition of the algorithm is imposed by a maximum limit of iterations.

2.2 Population Based Incremental Learning (PBIL)

The basis for introducing learning in GA's were established with the Population Based Incremental Learning (PBIL) algorithm proposed by Baluja and Caruana (1995), [23]. This algorithm means another optimization approach different than the GAs, obtaining excellent results in certain problems, like [18], [19] and [20]. Besides, it has been the starting point for EDAs, [16] and [17], and a good example for the proposed method CAPM.

The PBIL algorithm, [23] is partially based on the Equilibrium Genetic Algorithm (EGA), [21], but with some improvements made. The authors present it like a mixture between an EC algorithm and a hillclimbing approach, [18].

The main concept of the PBIL algorithm is the substitution of the genetic population with a set of statistics representing the information about the individuals. Therefore, the selection and crossover operators are not needed anymore, and a probability distribution process is the responsible for changing the populations each iteration.

The success of this statistic approach opened a wide research field with lot of works, [24], [16], [17], [26], etc..

Step 1 - Initialize probability matrix: this procedure is responsible for initializing randomly all the individuals, also called solution vectors. For this task

the probability matrix is firstly initialized with uniform distribution, equation 1, and all the alphabet elements will have the same likelihood of appearing.

$$P_{i=[0..(k-1)],j=[0..(L-1)]} = \frac{1.0}{k} \tag{1}$$

where k represents the alphabet size and L the chromosome size.

Step 2 - Evaluate and generate N vectors of solution: this procedure deals with the evaluation of the solution vectors. The solution vectors are generated after the first iteration through the sampling of the probability matrix. Once the best solutions are selected, they are used to update the information stored in the probability matrix.

Step 3 - Update the probability matrix: this step uses the following updating rule:

$$P_i(X = j) = (P_i(X = j) \times (1.0 - \alpha)) + (\alpha \times M_{i,j}) \tag{2}$$

where $P_i(X = j)$ is the probability of generate any of the j values which belong to the alphabet in the i^{th} position of the chromosome. α is the learning factor, and $M_{i,j}$ the probability matrix, column i, row j.

The learning factor of the algorithm (α) can differ depending on the problem. Besides, it affects the final results, [23]. Smaller learning rates imply wider searches, and higher values mean deeper search processes. However, its value should not be too high, because as the learning factor increases, the dependency between the solution and the initial population is higher.

Step 4 - Mutation of the probability matrix: the mutation operator plays an important role during the search process in order to guarantee the convergence avoiding local optimums and maintaining the diversity through the iterations.

The mutation operator in PBIL algorithms can be done at two levels: solution vector or probability matrix. Both of them are useful for maintaining the genetic diversity, but after several experiments made in different works, the probability matrix mutation appears to be slightly better, [22] or [23].

Step 5 - Repeat until the final condition: like the SGA and the FPGA the stop criterion is imposed by a maximum limit of iterations which depend on the problem to solve.

2.3 Chromosome Appearance Probability Matrix

In PBIL algorithms, the recombination operator is replaced by a vector of independent probabilities of each variable, and sampling this vector implies the study of the selections made by the algorithm till that moment. This concept, applied to IEC can be done in order to speed up the evolution in regards to the user needs. This was the key motivation for developing this new method based on the Chromosome Appearance Probability Matrix.

The steps of the proposed algorithm are explained in detail in [28] and [1], however this method introduces the following new features which differ from a canonical genetic algorithm.

Probability matrix for guiding mutation: when the user selects an element of the population, his or her selection is usually based on the collective combination of features in each element of an individual. For example, if the user is searching for tables he will appreciate the combination of several characteristics such as the color of legs of the table, the number, the shape of the surface, etc.. Therefore the information about the whole chromosome should be kept. To do this, a multidimensional array, with the same number of dimensions as genes, has been included. The bounds of the dimensions are determined by the number of alleles of the different genes.

The probability array 'M' is initialized by $M(gene_1, gene_2, gene_3, gene_m) = 1/T$ where 'm' is the number of genes, and gene_i could have values in $[allele_1^i, allele_2^i \ldots allele_{n_i}^i]$, and n_i the number of alleles of gene 'i'. The total possible combinations of chromosomes 'T' is calculated by multiplying the maximum sizes of each gene ($T = \prod_{i=1}^{m} n_i$).

This array shows the probability of being chosen that each possible combination of alleles have. Each iteration implies a selection of one or two individuals, and its chromosomes represent a position in the above array. After the selection, the corresponding position in the array is updated by a factor of α with the increment factor of the update rule, Δ_M. This Δ_M is calculated by the following equation:

$$\Delta_M = [M_{gene_s^1,\cdots,gene_s^n} \times (1.0 + \alpha))] - M_{gene_s^1,\cdots,gene_s^n} \quad (3)$$

The example in figure 1 shows how the update rule works for 1 chromosome with 2 different genes, gen_1 with 4 alleles {pos1,..,pos4}, and gen_2 with 10, {0..9}. It can be clearly seen how the probability matrix 'M' is updated with $\alpha = 0.005$ and how it affects to the rest cells.

The update operations take care that the sum of all the elements of the array will be 1. This array is very useful to keep information about the selection frequency of a determined chromosome, and therefore, to help the mutation process to evolve towards the preferences of the user.

Oriented mutation operator: the mutation operator is responsible for the mutation of the individuals. Once a gene has been selected for mutation, a specific chromosome is taken as the base of the mutation process (reference chromosome). This chromosome is selected from the whole possible chromosomes following a uniform distribution fixed by the probability array. The higher the value of a chromosome's likelihood array, the better the probability of being chosen. In the mutation process, a position in the chromosome to be mutated is randomly selected. Then, the gene in this position is substituted by a gene from the reference chromosome in that same position. Thus, the mutation operator is the result of a function from the chromosome to be mutated and the reference chromosome:

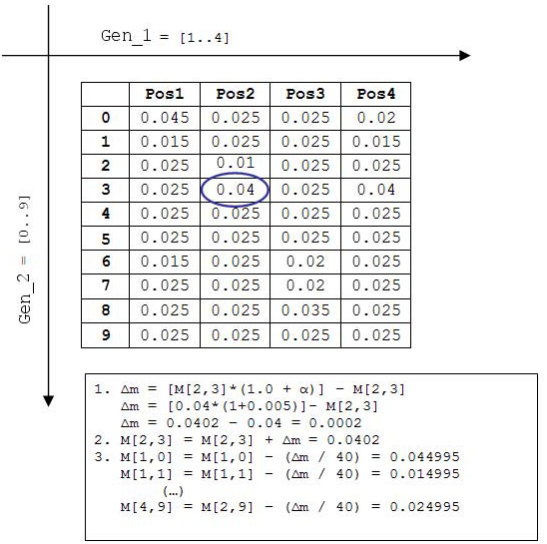

Fig. 1. Update rule example for CAPM algorithm, $\alpha = 0.005$

$$newChrom' = mutateOneGen(OldChrom, referenceChrom) \tag{4}$$

This approach has the ability to broadcast the preferences of the user towards all the chromosomes of the individuals.

The inclusion of a clone remover operator: the clone remover operator is responsible for mutating all those individuals which have exactly the same genetic structure as the other individuals in the same population.

Replacement of all the population but parents with the new individuals: the proposed strategy is elitist, however, as the user is not interested in evaluating the same individuals between iterations, the algorithm mutates the parents for the next generation, making them slightly different.

3 Experimental Tests

3.1 Trademark Finder

The problem arises from an idea proposed in [25] and it is explained more in detail in [1]. The challenge of the trademark finder is to help the user in the task of finding a specific logo which is new, different, or eye-catching, for a product, or a company. For this purpose, like in brainstorming process, the system offers different types of words, applying different colors, backgrounds, and styles. This

is only a first version for making the experiments, but in future developments figures and letter positions should change.

The algorithm starts randomly with six words (population), each one made of five letters, with random colors and styles. The user must select in each iteration the best two alternatives for him. Thus, each word is an individual of the population, and each letter is a chromosome with four genes representing color, font type, size and background, respectively. The search space is $3,2 \times 10^{21}$.

3.2 Experimental Framework

Numerous experiments (≥ 100.000) are required to validate empirically the efficiency of the methods. It must be taken into account that the problem should be run repeatedly for all the possible parameters and algorithms. Besides the fitness function must be the same for all the experiments in order to do an homogeneous comparative of the results. This is impossible to achieve with users (individual subjective preferences). Therefore, it is necessary to develop a simulator which replaces the human interaction acting like a virtual user equal for all the experiments, no matter the algorithm tested.

Evaluation based on ranking of preferences: for some type of products and business the preferences of corporate colors and fonts can be clear at the beginning of the search. Perhaps all letters in red are preferred, but the type of red can be selected by the user. For this reason, in the first test, an automatic evaluation which simulates the user behavior was designed with a ranking of preferences evaluation function.

However, the problem with this first test is that the automatic evaluation function is a non real world based function. To make the problem more complex it has been decided to include more realistic conditions which means non linear evaluation function and fluctuation in the preferences of the virtual user.

Evaluation based on fluctuating decisions rules sets: like humans do, it is possible that the user could find more than one type of solution interesting. Furthermore, after the study of the experiments conducted with humans [25] it was realized that very often the user does not have a clear idea of his own preferences. Also, often happened that his preferences or criteria changed as new designs were proposed. To simulate those doubts inherent to human behaviours, two sets of predefined evaluation rules which confront different preferences were included. Those rules randomly change with a probability of 70% using set 1, and 30% using set 2 (each iteration). These rules are applied independently to each chromosome (character) to obtain an objective measure.

In order to make even harder the search two conditions were included. These conditions forced the solution to search words with the same background for first and last letter and different for the remainder.

3.3 Comparative Study

This section presents a comparative study of some simulations in order to show the behavior of the algorithms explained in section 2. In the experiments devel-

oped the final condition is reached when the fitness of one element is 100 points (optimal solution), or when the iterations surpass 50. A valid measure is consider when the fitness is greater than 90.

At this point, after making 10.000 experiments per algorithm, running them with different parameters, see table 1, and both types of automatic evaluation function, the following results were achieved (best parameters found for each algorithm):

For the evaluation function based on the ranking the preferences only the CAPM method reaches the optimal solution in all experiments done (iteration 32 in average), and neither of the others even reached a valid solution.

When comparing the results, there is a significant difference between the algorithms: CAPM reaches much faster solutions in terms of iterations and better quality solutions. As can been seen clearly in tables 1, 2 and also in figure 2.

Besides, the PBIL shows clearly a low performance when working with micropopulations, due to the genetic diversity during the first 50 iterations is too low. Experiments with mutation likelihood of 45% have been run but this high mutation rate does not help the algorithm and makes the search completely random.

The FPGA has a good performance, but does not always beat the SGA results. The main reason is that the Euclidean distance equation used for the fitness prediction is not useful when predicting ranking evaluations or fluctuating rule sets. At this point future developments to try FPGA with prediction functions based on Neural Networks or Support Vector Machines are suggested.

Looking at the results for the fluctuating decisions evaluation model, the user must take at least 35 iterations in average to obtain a valid solution, and never reaches an optimal solution. However, the results are no so bad taking into account that the evaluation is using a non-linear function which changes of evaluation rules randomly (with 30% of probabilities of changing each iteration).

Table 1. Parameters used for experiments

Algorithm	Population Size	Mutation Prob.	Learning factor (α)
SGA	10	$[5\% \dots 45\%]$	N/A
FPGA	10(100)	$[5\% \dots 45\%]$	N/A
PBIL	10	$[5\% \dots 45\%]$	$[0,01 \dots 0,1]$
CAPM	10	$[5\% \dots 45\%]$	$[0,001 \dots 0,01]$

Table 2. Results with ranking of preferences

Iteration	SGA_10	$FPGA\_100$	$PBIL\_10$	$CAPM\_10$
10	50,00	51,00	30,00	62,00
20	65,00	65,00	35,00	85,00
30	74,00	74,00	38,00	**98,00**
40	80,00	80,00	43,00	**100(32)**
50	85,00	84,50	49,00	**100**

Table 3. Results with fluctuating decisions rule sets

Iteration	SGA_10	$FPGA\_100$	$PBIL\_10$	$CAPM\_10$
10	51,25	52,00	35,50	58,00
20	63,50	61,50	37,00	71,50
30	70,00	70,00	39,50	84,50
40	76,00	74,50	40,50	**91,50**
50	80,00	77,99	44,50	**96,00**

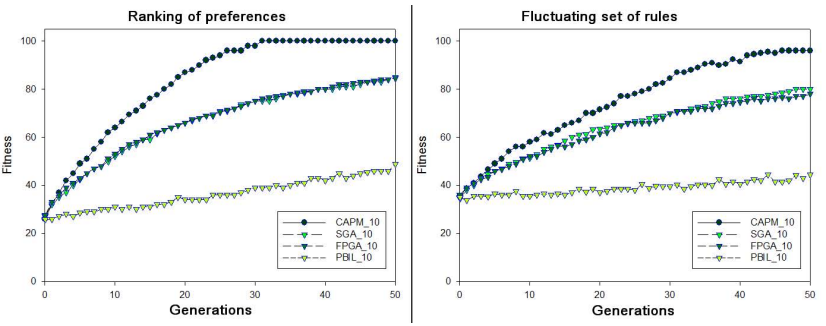

Fig. 2. Average fitness per iteration in both evaluation functions

It applies opposite evaluation preferences for each set of rules, and finally finds valid solutions after 35 iterations in average. The evolve process is very complex and is unpredictable what is really going to happen with the user. It is not usual to see the user changing completely of preferences each iteration (with 30% of probabilities), but it was decided to experiment with the worst case. In fact, in this evaluation mode all the algorithms tested except CAPM are unable to find even a valid solution (fitness ≥ 90).

4 Conclusion

After the study and development of different algorithms and their formal comparative test for solving an specific problem of IEC, the following conclusions have been drawn:

1. To test formally algorithms in IEC frameworks it is necessary to run a large number of experiments. This task can only be achieved when working with automatic evaluation functions. These functions must simulate human characteristics, which are not always predictable. For this paper the simulator changes its opinions or preferences randomly, with a probability of 30%. Also, it has been decided to make the search based on a non linear fitness function.

2. The analysis of the results of the SGA, shows that, although it is considered sufficient so far, it is not good enough for IEC, because it never reaches a valid solution in the experiments made. Besides, the proposed method has been compared successfully with an algorithm specifically developed for solving IEC problems (FPGA) and with a probabilistic algorithm (PBIL).
3. The proposed method has the capability of learning by the study of the selections made by the user. This alternative improves the results given by the SGA, FPGA and PBIL in terms of the quality of the solutions and the number of iterations in finding valid/optimal solutions.

   Finally, as the micropopulations affects negatively to all algorithms tested, becoming the main reason to decrease their performance, to avoid this problem a proposal is made to increase the selection pressure with the probability matrix. However, other proposals based on predictive fitness must be studied too.

Acknowledgment. This article has been financed by the Spanish founded research MCyT project OPLINK, Ref: TIN2006-08818-C04-02.

References

1. Sáez Y., Isasi P., Segovia J., J.C. Hern'andez "Reference chromosome to overcome user fatigue in IEC," *New Generation Computing, vol. 23, number 2, Ohmsha - Springer, pp. 129–142* (2005).
2. Goldberg D. E., Deb K., and Clark J. H., "Genetic algorithms, noise, and the sizing of populations," *Complex Syst., vol. 6, no. 4*, pp. 333-362. (1992).
3. Harik G., Cantú-Paz E., Goldberg D. E., and Miller B. L., "The Gambler's ruin problem, genetic algorithms, and the sizing of populations," *Transactions on Evolutionary Computation, vol. 7*, pp. 231–253, (1999).
4. Goldberg D.E., and Rundnick M. "Genetic algorithms and the variance of fitness", *Complex Systems, vol. 5, no. 3*, pp. 265–278, (1991).
5. J. He and X. Yao, "From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation, vol. 6*, pp. 495–511, Oct. (2002).
6. Sims K., "Artificial Evolution for Computer Graphics," *Comp. Graphics,Vol. 25, 4*, pp. 319–328. (1991).
7. Moore, J.H. "GAMusic: Genetic algorithm to evolve musical melodies." *Windows 3.1 Software available in: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/gamusic/0.html.* (1994).
8. Graf J., Banzhaf W. "Interactive Evolutionary Algorithms in Design. Procs of Artificial Neural Nets and Genetic Algorithms, *Ales, France*, pp. 227–230, (1995).
9. F.J. Vico, F.J. Veredas, J.M. Bravo, J. Almaraz "Automatic design sinthesis with artificial intelligence techniques." *Artificial Intelligence in Engineering 13*, pp. 251–256, (1999).
10. Santos A., Dorado J., Romero J., Arcay B., Rodríguez J. "Artistic Evolutionary Computer Systems," *Proc. of the GECCO Workshop, Las Vegas.* (2000).
11. Unemi T. "SBART 2.4: an IEC Tool for Creating 2D images, movies and collage, Proc. of the Genetic and Evolutionary Computation" *Conference Program, Las Vegas.* (2000).

12. Rowland D. "Evolutionary Co-operative Design Methodology: The genetic sculpture park." *Proc. of the GECCO Workshop, Las Vegas.* (2000).
13. Bentley P., "From Coffee Tables to Hospitals: Generic Evolutionary Design," *Evolutionary design by computers, Morgan-Kauffman*, pp. 405–423. (1999).
14. Takagi H. "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation," *Proc. of the IEEE, vol 89, number 9*, pp. 1275–1296 (2001).
15. Hsu F.-C. and Chen J.-S., "A study on multi criteria decision making model: Interactive genetic algorithms approach," *IEEE Int. Conf. on System, Man, and Cybernetics (SMC99)*, pp. 634–639 (1999).
16. Larrañaga P. and Lozano J. A., "Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation." *Boston, MA: Kluwer*, (2001).
17. Larrañaga P. and Lozano J. A., Bengoetxea E. "Estimation of distribution algorithms based on multivariate normal and Gaussian networks" *KZZA-IK-1-01*, (2001).
18. Baluja S., "An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics" *CMU-CS-95-193*, (1995).
19. Sebag M. and Ducoulombier A., "Extending Population-Based Incremental Learning to Continuous Search Spaces" *Lecture Notes in Computer Science", vol 1498*, pp. 418–426 (1998).,
20. Monmarché N. and Ramat E. and Dromel G. and Slimane M. and Venturini G., "On the similarities between AS, BSC and PBIL: toward the birth of a new metaheuristic" *citeseer.ist.psu.edu/557467.html*, (1999).
21. Juels A. and Baluja S. and Sinclair A., "The Equilibrium Genetic Algorithm and the Role of Crossover", *citeseer.ist.psu.edu/juels93equilibrium.html*, (1993).
22. Juels A. and Wattenberg M., "Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms", *Department of Computer Science, University of California at Berkeley*, (1995).
23. Baluja S., Pomerleau D., Jochem T.", "Towards Automated Artificial Evolution for Computer-generated Images" *Connection Science* 325–354, (1994).
24. Gonzalez C., Lozano J., Larranarraga P. "Analyzing the PBIL algorithm by means of discrete dynamical systems.", *Complex Systems.*, (1997)
25. Sáez Y., Sanjuán O., Segovia J., Isasi P., "Genetic Algorithms for the Generation of Models with Micropopulations," *Proc. of the EUROGP'03, Univ. of Essex, UK.* Apr. (2003).
26. Kern S., Muller S.D., Hansen N., Buche D., Ocenasek J., Koumoutsakos P. "Learning Probability Distributions in Continuous Evolutionary Algorithms, A Comparative Review"., *Kluwer Academic Publishers*, (2004).
27. Nishio K., Murakami M., Mizutani E., Honda N. "Efficient fuzzy fitness assignment strategies in an interactive genetic algorithm for cartoon face search"., *In Proc. Sixth International Fuzzy Systems Association World Congress (IFSA'95), pp. 173–176*, (2005).
28. Y. Saez and P. Isasi and J. Segovia, "Interactive Evolutionary Computation algorithms applied to solve Rastrigin test functions," *In Proc. of Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST 05), Springer-Verlag, pp. 682–691* (2005).